

# A Globally and Superlinearly Convergent Modified BFGS Algorithm for Unconstrained Optimization

Yaguang Yang\*

December 27, 2012

## Abstract

In this paper, a modified BFGS algorithm is proposed. The modified BFGS matrix estimates a modified Hessian matrix proposed in [17], which is a convex combination of an identity matrix for the steepest descent algorithm and a Hessian matrix for Newton's algorithm. The coefficient of the convex combination in the modified BFGS algorithm is dynamically chosen in every iteration. It is proved that, for any twice differentiable nonlinear function (convex or non-convex), the algorithm is globally convergent to a stationary point. If the stationary point is a local minimizer where the Hessian is strongly positive definite in a neighborhood of the minimizer, the iterates will eventually enter and stay in the neighborhood, and the modified BFGS algorithm reduces to the BFGS algorithm in this neighborhood. Therefore, the modified BFGS algorithm is superlinearly convergent. Moreover, the computational cost of the modified BFGS in each iteration is almost the same as the cost of the BFGS. Numerical test on the CUTE test set is reported. The performance of the modified BFGS algorithm implemented in our Matlab function `mBFGS` is compared to the BFGS algorithm implemented in the Matlab Optimization Toolbox function `fminunc`, a limited memory BFGS implemented as `L-BFGS`, a descent conjugate gradient algorithm implemented as `CG-Descent 5.3`, and a limited memory, descent and conjugate algorithm implemented as `L-CG-Descent`. This result shows that the modified BFGS algorithm may be very effective.

**Keywords:** modified BFGS algorithm, global convergence, superlinear convergence, nonconvex unconstrained optimization.

## 1 Introduction

The BFGS algorithm is one of the most successful algorithms for unconstrained nonlinear programming [2]. Although global and superlinear convergence results have been established for convex problems [13], it has been proved that, for general problems, the BFGS algorithm with Wolfe line search may not be convergent for nonconvex nonlinear functions [4]. Unfortunately, Wolfe line search condition is one of the prerequisites for applying the Zoutendijk theorem [18] to prove the global convergence of optimization algorithms. This motivates us to find a modified BFGS algorithm that is globally convergent for all twice differentiable nonlinear functions, convex or nonconvex. We also would like the behavior of the modified BFGS algorithm to be the same as the BFGS algorithm when the iterates approach a minimizer where the strong second order condition is met, i.e., we would like the proposed algorithm to be locally superlinearly convergent.

We will first examine how a modified Newton algorithm [17] achieves global and quadratic convergence. It uses a modified Hessian matrix which is a convex combination of the Hessian for Newton's algorithm and the identity matrix for the steepest descent algorithm. The most obvious advantage of using the convex combination other than linear combination of these matrices is that the modified Newton algorithm may take the steepest descent iteration or Newton's iteration; it has the merits of both the steepest descent algorithm and Newton's algorithm, i.e., it is globally and quadratically convergent. Similar to the idea that the BFGS estimates the Hessian matrix, we propose a modified BFGS update that estimates the

---

\*NRC, Office of Research, 21 Church Street, Rockville, 20850. Email: yaguang.yang@verizon.net

modified Hessian matrix given in [17]. We will prove that this modified BFGS algorithm is indeed globally convergent, as the modified Newton algorithm is. In addition, if the limit point is a local minimizer and its Hessian is strongly positive definite in a neighborhood of the minimizer, we will show that the iterates will enter the neighborhood and the modified BFGS will reduce to the BFGS, therefore, the modified BFGS is locally superlinearly convergent as the BFGS is.

The proposed modified BFGS update is different from other modified BFGS updates such as [9] and [14] in several aspects. First, our modified BFGS algorithm may take the steepest descent direction, while other modified BFGS algorithm may not. Second, the selection of the parameter of the convex combination is different from other methods.

The modified BFGS is implemented in the Matlab function `mBFGS`. The implementation `mBFGS` and an implementation of BFGS in Matlab Optimization Toolbox `fminunc` are tested against the CUTE test set [1] [3]. The performance of `mBFGS` is compared to `fminunc`, and other established and/or state-of-the-art optimization software, such as a limited memory BFGS algorithm [11] implemented as `L-BFGS`, a descent conjugate gradient algorithm [8] implemented as `CG-Descent 5.3`, and a limited memory descent and conjugate algorithm [7] implemented as `L-CG-Descent`. This result shows that the modified BFGS may be very effective.

The remainder of the paper is organized as follows. Section 2 introduces the modified BFGS algorithm. Section 3 discusses the algorithm's convergence properties. Section 4 provides the test results. Section 5 presents the conclusions.

## 2 The Modified BFGS Method

Our objective is to minimize a multi-variable nonlinear (convex or nonconvex) function

$$\min f(x), \quad (1)$$

where  $f$  is twice differentiable and  $x \in \mathbf{R}^n$ . Throughout the paper, we define by  $g(x)$  or simply  $g$  the gradient of  $f(x)$ , by  $H(x)$  or simply  $H$  the Hessian of  $f(x)$ . We denote by  $H \succ 0$  if a matrix  $H$  is positive definite, by  $H \succeq 0$  if  $H$  is positive semidefinite. We will use subscript  $k$  for the  $k$ th iteration, hence,  $x_0$  is used to represent the initial point. Denote by  $\bar{x}$  a local minimizer of (1), then

$$g(\bar{x}) = 0. \quad (2)$$

We make the following assumptions in the convergence analysis.

### Assumptions:

1. For an open set  $\mathcal{M}$  containing the level set  $\mathcal{L} = \{x : f(x) \leq f(x_0)\}$ ,  $g(x)$  is Lipschitz continuous, i.e., there exists a constant  $L > 0$  such that

$$\|g(x) - g(y)\| \leq L\|x - y\|, \quad (3)$$

for all  $x, y \in \mathcal{M}$ .

2. There are positive numbers  $\delta > 0$ ,  $1 > m > 0$ ,  $M > 1$ , and a neighborhood of  $\bar{x}$ , defined by  $\mathcal{N}(\bar{x}) = \{x : f(x) - f(\bar{x}) \leq \delta\}$ , such that for all  $x \in \mathcal{N}(\bar{x})$  and for all  $z \in \mathbf{R}^n$ ,

$$m\|z\|^2 \leq z^T H(x) z \leq M\|z\|^2. \quad (4)$$

3. There is a positive number  $L > 0$  such that for all  $x \in \mathcal{N}(\bar{x})$ ,

$$\|H(x) - H(\bar{x})\| \leq L\|x - \bar{x}\|. \quad (5)$$

Assumption 1 is required when we use the Zoutendijk theorem to establish the global convergence for the modified BFGS algorithm. Assumption 2 indicates that for all  $x \in \mathcal{N}(\bar{x})$ , a strong second order sufficient condition holds, i.e., there is a unique minimum in the neighborhood  $\mathcal{N}(\bar{x})$ , which will be used to prove the global and superlinear convergence.  $m$  and  $M$  are also used to choose the coefficient of the convex

combination. Assumption 3 will be needed only for the proof of the superlinear convergence. The  $L$  in (3) may be different from the  $L$  in (5). But we can always choose the largest value of  $L$  so that (3) and (5) will hold for the same  $L$ , which will simplify our notation.

For most optimization algorithms, the search for the minimizer of (1) is carried out by using

$$x_{k+1} = x_k + \alpha_k d_k, \quad (6)$$

where  $\alpha_k$  is the step size, and  $d_k$  is the search direction. For Newton's method, the search direction  $d_k$  is defined by

$$H(x_k)d_k = -g(x_k),$$

and the step size is set to  $\alpha_k = 1$ . If Newton's method converges, it converges fast (quadratic) but it may not converge at all and the computation of  $H(x_k)$  is expensive. The BFGS algorithm is developed to reduce the cost of the computation of  $H(x_k)$  while retaining the feature of fast (superlinear) convergence. It estimates  $H(x_k)$  using the following update formula

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}, \quad (7)$$

where

$$s_k = x_{k+1} - x_k, \quad y_k = g(x_{k+1}) - g(x_k). \quad (8)$$

On the other hand, since Newton's algorithm may not converge, modified Newton algorithms are introduced. In a modified Newton algorithm proposed in [17], a modified Hessian  $(\gamma_k I + (1 - \gamma_k)H(x_k))$  is suggested and the search for minimizer is carried out along a direction  $d_k$  that satisfies

$$(\gamma_k I + (1 - \gamma_k)H(x_k))d_k = Q(x_k)d_k = -g(x_k), \quad (9)$$

where  $\gamma_k \in [0, 1]$  is carefully selected in every iteration. Clearly, the modified Hessian  $Q(x_k)$  is a convex combination of the identity matrix for the steepest descent algorithm and the Hessian for Newton's algorithm. When  $\gamma_k = 1$ , the algorithm reduces to the steepest descent algorithm; when  $\gamma_k = 0$ , the algorithm reduces to Newton's algorithm. The global and quadratic convergence for the modified Newton algorithm is established in [17].

Since the iteration of the modified Newton algorithm is expensive because of the computation of  $H(x_k)$  and selection of  $\gamma_k$  which involves the computation of the smallest and the largest eigenvalues of  $H(x_k)$ , and the BFGS algorithm may not converge, we propose using a modified BFGS to estimate the modified Hessian. We show that the computational cost of the modified BFGS in each iteration is roughly the same as computational cost of the BFGS, and the modified BFGS algorithm is globally and superlinearly convergent.

Note that the BFGS update  $B_{k+1}$ , which is an estimation of  $H(x_{k+1})$ , is derived from secant equation

$$y_k = B_{k+1}s_k. \quad (10)$$

From (9) and (10), the modified BFGS update  $E_{k+1}$ , which is an estimation of  $Q(x_{k+1}) = \gamma_k I + (1 - \gamma_k)H_{k+1}$ , is suggested to satisfy

$$z_k = E_{k+1}s_k = (\gamma_k I + (1 - \gamma_k)B_{k+1})s_k = \gamma_k s_k + (1 - \gamma_k)y_k, \quad (11)$$

where  $\gamma_k \in [0, 1]$  will be carefully selected in every iteration. If  $\gamma_k = 1$ ,  $E_{k+1} = I$  estimates  $Q(x_{k+1})$  and the modified BFGS reduces to the steepest descent method from (9). If  $\gamma_k = 0$ ,  $E_{k+1} = B_{k+1}$  estimate  $Q(x_{k+1}) = H_{k+1}$  and the modified BFGS reduces to the BFGS method.

It is straightforward to derive the modified BFGS formula from (11) following exactly the same procedures of [12, pages 197-198], i.e.,

$$E_{k+1} = E_k - \frac{E_k s_k s_k^T E_k}{s_k^T E_k s_k} + \frac{z_k z_k^T}{z_k^T s_k}. \quad (12)$$

Using the Sherman-Morrison-Woodbury formula [5, page 51], we have

$$E_{k+1}^{-1} = \left( I - \frac{s_k z_k^T}{z_k^T s_k} \right) E_k^{-1} \left( I - \frac{z_k s_k^T}{z_k^T s_k} \right) + \frac{s_k s_k^T}{z_k^T s_k}. \quad (13)$$

Therefore, the modified BFGS search direction is defined by

$$E_k d_k = -g_k, \quad (14)$$

but the search direction  $d_k$  is calculated by

$$d_k = -E_k^{-1} g_k, \quad (15)$$

where  $E_k^{-1}$  is updated using (13). To apply the Zoutendijk theorem in the global convergence analysis in the next section,  $d_k$  is desired to be a descent direction, this requires  $E_k \succ 0$  for all  $k \geq 0$ . Assume  $E_0 \succ 0$  is selected, and  $E_{k-1} \succ 0$  is obtained. Since  $d_{k-1}$  is a descent direction from (14),  $s_{k-1} = x_k - x_{k-1} \neq 0$ . If we can select  $\gamma_{k-1}$  such that  $z_{k-1}^T s_{k-1} > 0$ , then from (13), it is easy to check that for any  $0 \neq v \in \mathbf{R}^n$ ,  $v^T E_{k-1}^{-1} v > 0$ . Therefore,  $E_k \succ 0$ , i.e.,  $d_k$  is indeed a descent direction for all  $k \geq 0$ . As a matter of fact, we want to select  $\gamma_k$  to meet the stronger conditions

$$m \leq \frac{z_k^T s_k}{s_k^T s_k} \quad \text{and} \quad \frac{z_k^T z_k}{z_k^T s_k} \leq M, \quad (16)$$

where  $m$  and  $M$  satisfy  $0 < m < 1 < M < \infty$ . (16) will be used to prove the global convergence of the modified BFGS algorithm. The first inequality of (16) can be rewritten as

$$\begin{aligned} z_k^T s_k &= (\gamma_k s_k^T + (1 - \gamma_k) y_k^T) s_k \geq m s_k^T s_k \\ \iff \gamma_k (s_k^T s_k - y_k^T s_k) &\geq m s_k^T s_k - y_k^T s_k. \end{aligned}$$

Denote

$$\tilde{\gamma}_k = \frac{m s_k^T s_k - y_k^T s_k}{s_k^T s_k - y_k^T s_k}. \quad (17)$$

Since  $\gamma_k \in [0, 1]$ , the first inequality of (16) is equivalent to

$$\begin{cases} \max\{\tilde{\gamma}_k, 0\} \leq \gamma_k \leq 1 & \text{if } s_k^T s_k > y_k^T s_k, \\ 0 \leq \gamma_k \leq 1, & \text{if } s_k^T s_k = y_k^T s_k, \\ 0 \leq \gamma_k \leq 1 \leq \tilde{\gamma}_k, & \text{if } s_k^T s_k < y_k^T s_k. \end{cases} \quad (18)$$

The second inequality of (16) can be rewritten as

$$z_k^T z_k = (\gamma_k s_k + (1 - \gamma_k) y_k)^T (\gamma_k s_k + (1 - \gamma_k) y_k) \leq M (\gamma_k s_k + (1 - \gamma_k) y_k)^T s_k \quad (19a)$$

$$\iff p(\gamma_k) \equiv \gamma_k^2 (s_k - y_k)^T (s_k - y_k) + \gamma_k (s_k - y_k)^T (2y_k - M s_k) + y_k^T (y_k - M s_k) \leq 0. \quad (19b)$$

$p(\gamma_k)$  is a quadratic and convex function of  $\gamma_k$ . Since  $M > 1$ , it is easy to see that the strict inequality of (19a) holds for  $\gamma_k = 1$ , hence  $p(1) < 0$ . Therefore  $p(\gamma_k) = 0$  has two solutions  $\underline{\gamma}_k$  and  $\bar{\gamma}_k$  satisfying  $\underline{\gamma}_k < 1 < \bar{\gamma}_k$  and for any  $\gamma_k \in [\underline{\gamma}_k, \bar{\gamma}_k]$ , (19a) holds. From (19b), if  $s_k \neq y_k$ ,

$$\begin{aligned} \underline{\gamma}_k &= \frac{(s_k - y_k)^T (M s_k - 2y_k) - \sqrt{((s_k - y_k)^T (M s_k - 2y_k))^2 - 4(s_k - y_k)^T (s_k - y_k) y_k^T (y_k - M s_k)}}{2(s_k - y_k)^T (s_k - y_k)} \\ &= \frac{(s_k - y_k)^T (M s_k - 2y_k) - \sqrt{(M(s_k^T (s_k - y_k))^2 + 4(M - 1)(s_k^T s_k y_k^T y_k - (y_k^T s_k)^2))}}{2(s_k - y_k)^T (s_k - y_k)}; \end{aligned} \quad (20)$$

if  $s_k = y_k$ , then, the inequality (19) holds for  $\gamma_k = 0$ . Therefore,

$$\underline{\gamma}_k < 0. \quad (21)$$

**Remark 2.1** (13) can be replaced by the following equivalent representation

$$E_{k+1}^{-1} = E_k^{-1} - \frac{s_k}{z_k z_k^T} E_k^{-1} - E_k^{-1} z_k \frac{s_k^T}{z_k z_k^T} + \frac{s_k}{z_k z_k^T} (z_k^T E_k^{-1} z_k) \frac{s_k^T}{z_k z_k^T} + \frac{s_k s_k^T}{z_k^T s_k}. \quad (22)$$

which requires fewer computational counts than (13) does. However, this equivalent formula is not numerically stable. For CUTE problem **heart6ls**, when the condition number of  $E_{k+1}^{-1}$  is poor, two of the eigenvalues of  $E_{k+1}^{-1}$  generated by (22) are negative, but all eigenvalues of  $E_{k+1}^{-1}$  generated by (13) are greater than zero.

**Remark 2.2** Although the two formulae in (20) are equivalent, the second one is numerically much more stable. For the CUTE test problem **djk1**, using the first formula results in a negative value inside the square root because of the computational error, while the second formula ensures a positive value inside the square root.

Since  $\gamma_k \in [0, 1]$ , therefore

$$\underline{\gamma}_k \leq \gamma_k \leq 1. \quad (23)$$

Intuitively, it is desirable to have  $\gamma_k \in [0, 1]$  as close to zero as possible so that the algorithm will approach to the standard BFGS algorithm. Therefore, we want to select the smallest  $\gamma_k \in [0, 1]$  satisfying (18) and (23). We consider all possible relations among  $ms_k^T s_k$ ,  $s_k^T s_k$ , and  $y_k^T s_k$ . Since  $m < 1$ , we have  $ms_k^T s_k < s_k^T s_k$ .

- Case 1 ( $y_k^T s_k < ms_k^T s_k < s_k^T s_k$ ): To select the smallest  $\gamma_k \in [0, 1]$  satisfying (18) and (23), combining the first relation of (18) and (23), we have  $\gamma_k = \max\{\max\{0, \check{\gamma}_k\}, \underline{\gamma}_k\}$ . From (17), we know  $\check{\gamma}_k > 0$  in this case. Therefore,  $\gamma_k = \max\{\check{\gamma}_k, \underline{\gamma}_k\}$ .
- Case 2 ( $ms_k^T s_k \leq y_k^T s_k < s_k^T s_k$ ): To select the smallest  $\gamma_k \in [0, 1]$  satisfying (18) and (23), combining the first relation of (18) and (23), we have  $\gamma_k = \max\{\max\{0, \check{\gamma}_k\}, \underline{\gamma}_k\}$ . From (17), we know  $\check{\gamma}_k \leq 0$  in this case. Therefore,  $\gamma_k = \max\{0, \underline{\gamma}_k\}$ .
- Case 3 ( $ms_k^T s_k < s_k^T s_k \leq y_k^T s_k$ ): To select the smallest  $\gamma_k \in [0, 1]$  satisfying (18) and (23), combining the last 2 relations of (18) and (23), we have  $\gamma_k = \max\{0, \underline{\gamma}_k\}$ . In particular, when  $s_k = y_k$ , from (21),  $\underline{\gamma}_k < 0$ , therefore,  $\gamma_k = 0$ .

Combining all cases, we have

$$\gamma_k = \begin{cases} \max\{\underline{\gamma}_k, \check{\gamma}_k\}, & \text{if } ms_k^T s_k > y_k^T s_k, \\ \max\{0, \underline{\gamma}_k\}, & \text{if } ms_k^T s_k \leq y_k^T s_k, \\ 0, & \text{if } s_k = y_k. \end{cases} \quad (24)$$

**Remark 2.3** It is worthwhile to note that if  $y_k^T y_k \leq My_k^T s_k$  holds, then (19a) holds for  $\gamma_k = 0$ , i.e.,  $\underline{\gamma}_k \leq 0$ . In addition, if  $ms_k^T s_k \leq y_k^T s_k$  holds at the same time, then from (24),  $\gamma_k = 0$ . Moreover,  $d_k$  is a descent direction.

Now we are ready to present the modified BFGS algorithm.

#### Algorithm 2.1 Modified BFGS

Data:  $0 < \epsilon$ ,  $m < 1$ , and  $1 < M < \infty$ , initial  $x_0$ , and  $E_0 = I$ .

for  $k=0, 1, 2, \dots$

    Calculate gradient  $g(x_k)$ . If  $\|g(x_k)\| < \epsilon$ , stop;

    Compute search direction  $d_k$  using (15);

    Set  $x_{k+1} = x_k + \alpha_k d_k$ , where  $\alpha_k$  satisfies the Wolfe condition to be defined later;

    Compute  $s_k$  and  $y_k$  using (8);

Select  $\gamma_k$  using (24), and computer  $z_k$  using (11);

Update  $E_{k+1}^{-1}$  using (13);

$k \leftarrow k + 1$ ;

**end**

**Remark 2.4** *It is clear that the computation involving the selection of  $\gamma_k$  is negligible (requires  $\mathcal{O}(n)$  operations). Therefore, the cost of modified BFGS in each iteration is almost the same as the cost of the BFGS.*

In the rest of the paper, our discussion will focus on the proof of global and superlinear convergence of Algorithm 2.1. The convergence properties are directly related to the goodness of the search direction and step length. The quality of the search direction is measured by

$$\cos(\theta_k) = -\frac{g_k^T d_k}{\|g_k\| \|d_k\|} = \frac{d_k^T E_k s_k}{\|E_k d_k\| \|s_k\|}, \quad (25)$$

where the second equation follows from (6) and (14). A good step length  $\alpha_k$  should satisfy the following Wolfe condition.

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \sigma_1 \alpha_k g_k^T d_k, \quad (26a)$$

$$d_k^T g(x_k + \alpha_k d_k) \geq \sigma_2 g_k^T d_k, \quad (26b)$$

where  $0 < \sigma_1 < \sigma_2 < 1$ . The existence of the Wolfe condition is established in [15, 16]. An algorithm that finds, in finite steps, a point satisfying the Wolfe condition is given in [10]. Therefore, we will not discuss step size selection in this paper.

### 3 Convergence Analysis

An important global convergence result was given by Zoutendijk [18] which can be stated as follows.

**Theorem 3.1** *Suppose that  $f$  is bounded below in  $\mathbf{R}^n$  and that  $f$  is continuously twice differentiable in an open set  $\mathcal{M}$  containing the level set  $\mathcal{L} = \{x : f(x) \leq f(x_0)\}$ . Assume that the gradient is Lipschitz continuous on  $\mathcal{M}$ , i.e., there exists a constant  $L > 0$  such that*

$$\|g(x) - g(y)\| \leq L \|x - y\|, \quad (27)$$

for all  $x, y \in \mathcal{M}$ . Assume further that  $d_k$  is a descent direction and  $\alpha_k$  satisfies the Wolfe condition. Then

$$\sum_{k \geq 0} \cos^2(\theta_k) \|g_k\|^2 < \infty. \quad (28)$$

■

the Zoutendijk theorem indicates that if  $d_k$  is a descent direction and  $\cos(\theta_k) \geq \delta > 0$ , then the algorithm is globally convergent because  $\lim_{k \rightarrow \infty} \|g_k\| = 0$ .

We are now ready to state the main convergence result for the modified BFGS algorithm.

**Theorem 3.2** *Suppose that  $f$  is bounded below in  $\mathbf{R}^n$ ,  $f$  is continuously twice differentiable in an open set  $\mathcal{M}$  containing the level set  $\mathcal{L} = \{x : f(x) \leq f(x_0)\}$ , and Assumptions 1-3 hold. Then Algorithm 2.1 is globally convergent in the sense that  $\liminf \|g(x_k)\| \rightarrow 0$ . In addition, if  $s_k \rightarrow 0$ , then Algorithm 2.1 converges to some  $\bar{x}$  satisfying  $\|g(\bar{x})\| = 0$  with superlinear rate.*

**Proof:** First, we show that  $d_k$  is a descent direction. From the selection of  $\gamma_k$  using (24), we know that (16) holds. The first inequality of (16) guarantees  $z_k^T s_k > 0$ . Using this fact and (13), we conclude  $E_k \succ 0$  since  $E_0 = I \succ 0$ . Therefore,  $d_k$  is a descent direction. Since (16) holds for all  $k \geq 0$ , following exactly the same arguments used in the proof of [12, Theorem 8.5], we have a subsequence  $\{j_k\}$  such that

$$\cos(\theta_{j_k}) \geq \delta > 0. \quad (29)$$

In view of Theorem 3.1, (29) indicates

$$\liminf \|g(x_k)\| \rightarrow 0. \quad (30)$$

This means that there exists the  $\bar{x}$  and a subsequence of  $x_{j_k}$  such that  $x_{j_k} \rightarrow \bar{x}$  and  $\|g(\bar{x})\| = 0$ . Since the function is locally strongly convex in a neighborhood of every local minimizer  $\bar{x}$  with  $\|g(\bar{x})\| = 0$ , this means that every  $\bar{x}$  is isolated (there is a unique local minimizer in the neighborhood). Therefore, the condition  $s_k \rightarrow 0$  and (30) is enough to prove that  $x_k \rightarrow \bar{x}$ . Since Algorithm 2.1 is globally convergent, for sufficiently large  $k$ ,  $f(x_k) \leq f(\bar{x}) + \delta$ . Therefore, for all  $v \in \mathbf{R}^n$ ,

$$m\|v\|^2 \leq v^T H(x_k) v \leq M\|v\|^2. \quad (31)$$

Using Taylor's Theorem [12, Theorem 2.1]

$$y_k = g(x_{k+1}) - g(x_k) = \int_0^1 H(x_k + t\alpha_k d_k) s_k dt \equiv \bar{H}_k s_k. \quad (32)$$

(31) and (32) imply that  $\bar{H}_k$  is positive definite, i.e., for all  $v \in \mathbf{R}^n$ ,

$$m\|v\|^2 \leq v^T \bar{H}_k v \leq M\|v\|^2. \quad (33)$$

This gives

$$\frac{y_k^T s_k}{s_k^T s_k} = \frac{s_k^T \bar{H}_k s_k}{s_k^T s_k} = \frac{s_k^T}{\|s_k\|} \bar{H}_k \frac{s_k}{\|s_k\|} \geq m,$$

and

$$\frac{y_k^T y_k}{y_k^T s_k} = \frac{s_k^T \bar{H}_k^2 s_k}{s_k^T \bar{H}_k s_k} = \frac{(\bar{H}_k^{\frac{1}{2}} s_k)^T}{\|\bar{H}_k^{\frac{1}{2}} s_k\|} \bar{H}_k \frac{\bar{H}_k^{\frac{1}{2}} s_k}{\|\bar{H}_k^{\frac{1}{2}} s_k\|} \leq M.$$

From these two inequalities, in view of Remark 2.3, we conclude that for large  $k$ ,  $\gamma_k = 0$  is always selected. Therefore, the modified BFGS reduces to the standard BFGS for large  $k$ . In addition, if Assumption 3 holds, the BFGS converges at a superlinear rate, therefore the modified BFGS also converges at the rate because it is identical to the BFGS for large  $k$ . ■

## 4 Implementation and Numerical Test

This section provide detailed information about the implementation of the algorithm which is slightly different from the description of Algorithm 2.1. It also presents our test results against CUTE problems.

### 4.1 Implementation details

Algorithm 2.1 has been implemented in Matlab function `mBFGS` with the following considerations. First, the selection of  $m$  and  $M$  turns out to be important. The  $m$  and  $M$  of  $H(\bar{x})$  satisfying (4) depend on the individual function to be optimized and each of its local minimizers. To be safe, one may select small  $m$  and large  $M$ , which will be likely cover all possible functions to be optimized, but this selection may not be numerically stable. On the other hand, if  $m$  is selected too big, and/or  $M$  is selected too small, the selection may violate condition (4). Our selection of the default set of parameters are  $m = 0.00001$ ,  $M = 100000$ , and  $\epsilon = 0.00001$ , which, in general, give very impressive computational result.

Second, for several test problems, the condition numbers of the estimated  $E_k^{-1}$  are poor at the early iterations, which leads to very large vector  $d_k$ . If this happens, line search takes long time to find a better iterate. Therefore,  $d_k$  is re-scaled such that  $\|d_k\| = 10^6$  if  $\|d_k\| > 10^6$  is detected.

Test result on the algorithm with the above implementation is very impressive. However, for several problems, it takes many steps to converge, which may be due to the poor estimation of  $m$  and  $M$  by the default set of the parameters. Therefore, a modification that dynamically selects  $m_k$  and  $M_k$  is implemented in **mBFGS** for the purpose of getting better estimation of the bounds of a particular local minimizer of a particular function.

From (17) and (20), it is easy to see that  $m$  only affects the value of  $\tilde{\gamma}_k$ , and  $M$  only affects the value of  $\underline{\gamma}_k$ . We also noticed that  $m$  and  $M$  together affect the condition number of  $E_{k+1}$ , which is important to the numerical stability in the computation of  $d_{k+1}$  from (14). This requires the ratio of  $M$  to  $m$  as small as possible. On the other hand, we want to select  $m$  and  $M$  such that  $\tilde{\gamma}_k$  and  $\underline{\gamma}_k$  as small as possible to maximize the chance of using the BFGS formula. This requires the selection of small  $m$  and large  $M$ , or the selection of the ratio of  $M$  to  $m$  as large as possible. For the trade off, we select the ratio of  $M$  to  $m$  as  $10^{10}$ . The nominal parameters in **mBFGS** are  $\bar{m} = 0.00001$ ,  $\bar{M} = 100000$ , and  $\epsilon = 0.00001$ . Because of the fixed ratio of  $M$  to  $m$ , we must increase or decrease  $m$  and  $M$  at the same time. From (17) and (20), increasing  $m$  and  $M$  will decrease  $\underline{\gamma}_k$  but increase  $\tilde{\gamma}_k$ ; and decreasing  $m$  and  $M$  will increase  $\underline{\gamma}_k$  but decrease  $\tilde{\gamma}_k$ . From (24), we want the difference of  $\tilde{\gamma}_k$  and  $\underline{\gamma}_k$  to be small, so that the final choice  $\gamma_k$  is small. Therefore, we dynamically adjust  $m$  and  $M$  using the following simple heuristics.

```

Set  $m = \bar{m}$ 
 $M = \bar{M}$ 
Calculate  $\tilde{\gamma}_k$ 
if  $\tilde{\gamma}_k > 1$ 
    adjust  $M = 10^4 \bar{M}$ 
    then calculate  $\underline{\gamma}_k$ 
else
    calculate  $\underline{\gamma}_k$ 
    if  $\underline{\gamma}_k - \tilde{\gamma}_k > 0.2$  and  $\underline{\gamma}_k > 0$ 
        adjust  $M = 10^3 \bar{M}$  and  $m = 10^3 \bar{m}$ 
        then recalculate  $\underline{\gamma}_k$  and  $\tilde{\gamma}_k$ 
    elseif  $\tilde{\gamma}_k - \underline{\gamma}_k > 0.2$  and  $\tilde{\gamma}_k > 0$ 
        adjust  $M = 10^{-2} \bar{M}$  and  $m = 10^{-2} \bar{m}$ 
        then recalculate  $\underline{\gamma}_k$  and  $\tilde{\gamma}_k$ 
    end
end
end

```

The above modification significantly reduces the number of iterations for the problems which used many iterations if fixed  $m$  and  $M$  are used. Moreover, it has little impact for the remaining problems.

## 4.2 Numerical test

The modified BFGS implementation **mBFGS** and the BFGS algorithm implemented in Matlab Optimization Toolbox function **fminunc** are tested using the CUTE test problem set. **fminunc** options are set as

```
options = optimset('LargeScale','off','MaxFunEvals',1e+20,'MaxIter',5e+5,'TolFun',1e-20,'TolX',1e-10).
```

This setting is selected to ensure that the BFGS implementation **fminunc** will have enough iterations either to converge or to fail.

We conducted tests for both **mBFGS** and **fminunc** against the CUTE test problem set, which is downloaded from the Princeton test problem collections [1]. Since the CUTE test set is presented in AMPL mod-files, we first convert AMPL mod-files into nl-files so that Matlab functions can read the CUTE models, then we use Matlab functions **mBFGS** and **fminunc** to read the nl-files and solve these test problems. The objective function is calculated from AMPL command  $[f,c] = \text{amplfunc}(x,0)$ . The gradient



function is calculated from AMPL command  $[g, Jac] = \text{amplfunc}(x, 1)$ . Both **mBFGS** and **fminunc** use these values in the optimization algorithms. Because of the restriction of the conversion software which converts mod-files to nl-files, the test is done for all CUTE unconstrained optimization problems whose sizes are less than 300. The test uses the initial points provided by the CUTE test problem set, we record the calculated objective function values, the norms of the gradients at the final points, and the iteration numbers for these tested problems. We present this test results in Table 1. In this table, **iter** stands for the number of total iterations used by the algorithm; **obj** is the value of the objective function achieved at the end of the iterations; **gradient** is the norm of the gradient at the end of the iterations.

Table 1: Test result for problems in CUTE [3], initial points are given in CUTE

Problem	size n	iter mBFGS	obj mBFGS	gradient mBFGS	iter fminunc	obj fminunc	gradient fminunc
arglina	100	1	100.0	0.0	4	100.0	0.1662e-03
bard	3	18	0.8214e-02	0.2082e-06	20	0.8214e-02	0.1158e-05
beale	2	13	0.3075e-14	0.1817e-06	15	0.2400e-09	0.1392e-06
biggs6	6	70	0.1080e-09	0.7865e-05	68	0.4796e-03	<b>0.1802e-01</b>
box3	3	19	0.4077e-10	0.1828e-05	24	0.3880e-10	0.2364e-5
brkmcc	2	4	0.1690e-00	0.8114e-07	5	0.1690e-00	0.4542e-06
brownal	10	12	0.1406e-12	0.6968e-05	16	0.3050e-08	0.1043e-03
brownbs	2	632	0.1950e-17	0.5369e-06	11	0.9308e-04	<b>15798.5950e-00</b>
brownden	4	21	85822.2016e-00	0.3135e-03	32	85822.2017e-00	<b>0.4646e-00</b>
chnrosnb	50	158	0.1686e-15	0.9838e-05	98	30.0583e-00	<b>10.1863e-00</b>
cliff	2	27	0.1997e-00	0.8919e-05	1	1.0015e-00	<b>1.4147e-00</b>
cube	2	21	0.4231e-19	0.6905e-09	34	0.7987e-09	0.1340e-03
deconvu	51	80	0.3158e-06	0.1750e-03	80	0.3158e-06	0.1750e-03
denschna	2	7	0.1467e-13	0.3346e-06	10	0.5000e-12	0.1581e-05
denschnb	2	7	0.6047e-13	0.5505e-06	7	0.1000e-11	0.2200e-05
denschnc	2	8	<b>0.1833e-00</b>	0.4104e-07	21	0.1608e-08	0.3262e-03
denschnd	3	38	0.2461e-08	0.4040e-06	23	45.2971e-00	<b>84.5851e-00</b>
denschnf	2	10	0.4325e-17	0.3919e-07	10	0.2000e-10	0.1005028e-03
dixon3dq	10	15	0.5626e-15	0.5588e-07	20	0.1400e-11	0.3661e-5
djtl	2	79	-8951.5447e-00	0.2881e-01	3	-8033.8869e-00	<b>1273.3319e-00</b>
eigenals	110	79	0.7521e-13	0.3483e-05	78	0.1092e-02	<b>0.1029e-00</b>
eigenbls	110	513	0.1994e-10	0.7166e-05	91	0.3462e-00	<b>0.4642e-00</b>
engval2	3	27	0.1172e-12	0.9109e-95	29	0.3953e-09	0.2799e-03
errinros	81	48	0.2442e-96	0.5141e-95	92	0.4577e-03	<b>0.2553e-00</b>
expfit	2	11	0.2405e-00	0.2350e-05	12	0.2405e-05	0.2263e-05
extrosnb	10	1	0.0	0.0	1	0.0	0.0
fletcbv2	100	97	-0.5140e-00	0.9676e-05	98	-0.5140e-00	0.1087e-4
fletcher	100	179	0.1114e-13	0.4440e-05	63	68.1289e-00	<b>160.9879e-00</b>
genhumps	5	47	0.1871e-09	0.8571e-05	59	0.4493e-07	0.3167e-03
growthls	3	1	<b>3542.1490e-00</b>	0	12	12.4523e-00	<b>0.5809e-01</b>
hairy	2	18	20.0	0.5710e-05	22	20.0	0.3810e-04
hatfdd	3	23	0.6615e-07	0.1853e-05	19	0.066150e-07	0.2355e-05
hatfde	3	28	0.4434e-06	0.5321e-05	9	0.6210e-06	0.7970e-05
heart6ls	6	2180	0.2620e-14	0.6696e-5	53	0.6318e-00	<b>71.9382548e-00</b>
helix	3	22	0.5489e-16	0.1901e-06	29	0.2260e-10	0.4196e-04
hilberta	10	20	0.2422e-06	0.5563e-05	35	0.2289e-06	0.3263e-05
hilbertb	50	11	0.4606e-12	0.3123e-05	6	0.21e-11	0.6542e-5
himmelbb	2	1	0.9665e-13	0.1153e-06	6	0.1462e-04	0.1251e-02

himmelbf	2	7	0.1069e-12	0.9337e-06	8	0.1000e-12	0.1448e-05
himmelbg	2	7	0.1069e-12	0.9337e-06	8	0.1000e-12	0.1448e-05
himmelbh	2	7	-0.9999e-00	0.5188e-06	7	-0.9999e-00	0.2607e-06
humps	2	85	0.2281e-09	0.6755e-05	25	5.4248e-00	<b>2.3625e-00</b>
jensmp	2	1	<b>2020</b>	0	16	124.3621e-00	0.2897e-05
kowosb	4	27	0.3075e-03	0.1426e-05	33	0.3075e-03	0.1253e-06
loghairy	2	87	0.1823e-00	0.6892e-06	11	<b>2.5199e-00</b>	0.5377e-02
mancino	100	35	0.162e-16	0.9715e-05	9	0.2204e-02	<b>1.2243e-00</b>
maratosb	2	3	-1.0	0.5142e-07	2	-0.9997e-00	<b>0.3570e-01</b>
mexhat	2	1	-0.4009e-01	0.6621e-05	4	-0.4009e-01	0.13703e-04
osborneb	11	51	0.4013e-01	0.2474e-05	76	0.4013e-01	0.7884e-05
palmer1c	8	32	0.9759e-00	0.3995e-06	38	16139.4418e-00	<b>655.0159e-00</b>
palmer2c	8	86	0.1442e-01	0.1013e-05	60	98.0867e-00	<b>33.4524e-00</b>
palmer3c	8	47	0.1953e-01	0.8975e-05	56	54.3139e-00	<b>7.8518e-00</b>
palmer4c	8	77	0.5031e-01	0.6125e-05	56	62.2623e-00	<b>6.6799e-00</b>
palmer5c	6	12	2.1280e-00	0.2881e-05	14	2.1280e-00	0.7484e-03
palmer6c	8	55	0.1638e-01	0.4847e-05	43	18.0992e-00	<b>0.7851e-00</b>
palmer7c	8	40	0.6019e-00	0.1289e-05	28	56.9098e-00	<b>4.0268e-00</b>
palmer8c	8	46	0.1597e-00	0.4974e-05	49	22.4365e-00	<b>1.3147e-00</b>
powellsq	2	0	0	0	0	0	0
rosenbr	2	32	0.1382e-15	0.5153e-06	36	0.283e-10	2.6095e-5
sineval	2	68	0.1271e-15	0.9558e-06	47	0.2212e-00	<b>1.2315e-00</b>
sisser	2	14	0.2809e-07	0.8680e-05	11	0.1540e-7	0.7282671e-5
tointqor	50	37	1175.4722e-00	0.7734e-05	40	1175.4722e-00	0.9041e-07
vardim	100	21	0.0001e-20	0.1126e-07	1	0.2244e-06	<b>0.5511e-00</b>
watson	31	48	0.2442e-06	0.5141e-05	90	0.1050e-02	<b>0.4875e-00</b>
yfitu	3	74	0.6670e-12	0.5396e-05	57	0.4398e-02	<b>11.8427e-00</b>

We summarize the comparison of the test result as follows:

1. The modified BFGS function **mBFGS** converges in all the test problems after terminate condition  $\|g(x_k)\| < 10^{-5}$  is met except for three problems **brownden**, **deconvu**, and **djtl**. But for these problems, **mBFGS** finds better solutions than **fminunc**. Moreover, for about 40% of the problems, Matlab Toolbox BFGS function **fminunc** does not reduce  $\|g(x_k)\|$  to smaller than 0.01. For these problems, the objective functions obtained by **fminunc** normally are not close to the minimum;
2. For those problems that both **mBFGS** and **fminunc** converge, **mBFGS** most time uses less iterations than **fminunc** and converges to a point with smaller  $\|g(x_k)\|$ ;
3. There are three problems (**denschn**, **growthls**, and **jensmp**), **mBFGS** converges to a local minimum but **fminunc** finds a better point.

**Remark 4.1** We also tried (12) and (14) rather than (13) and (15) in the calculation of the search direction  $d_k$ . With this implementation, the algorithm converges with  $\|g(x_k)\| < 10^{-5}$  as required for all the test problems, including **brownden**, **deconvu**, and **djtl**. We noticed that although **mBFGS** stops before  $\|g(x_k)\| < 10^{-5}$  is achieved for these three problems, the objective functions obtained are essentially the same as if (12) and (14) are used. Given the fact that using (13) and (15) does not require solving the linear systems of equations but using (12) and (14) does, we suggest using the implementation described in this section.

Most of the above problems are also used, for example in [6], to test some established and state-of-the-art algorithms. In [6], 145 CUTEr unconstrained problems are tested against limited memory BFGS

algorithm [11] (implemented as **L-BFGS**), a descent and conjugate gradient algorithm [8] (implemented as **CG-Descent 5.3**), and a limited memory descent and conjugate gradient algorithm [7] (implemented as **L-CG-Descent**). The sizes of most of these test problems are smaller than or equal to 300. The size of the largest test problems in [6] is 10000. Since our AMPL conversion software does not work for problems whose sizes are larger than 300, we consider only problems [6] whose sizes are less than or equal to 300. We compare the test results obtained by our implementation of Algorithm 2.1 and the results obtained by algorithms [11, 8, 7] (reported in [6]). For this test, we changed the stopping criterion to  $\|g(x)\|_\infty \leq 10^{-6}$  for consistency. The test results are listed in Table 2.

Table 2: Comparison of mNewtow, L-CG-Descent, L-BFGS, and CG-Descent 5.3 for problems in CUTE [3], initial points are given in CUTE

Problem	size	methods	iter	nFun	nGrad	obj	gradient
arglina	200	mBFGS	1	14	2	1.000e+002	1.894e-015
		L-CG-Descent	1	3	2	<b>2.000e+002</b>	3.384e-008
		L-BFGS	1	3	2	<b>2.000e+002</b>	3.384e-008
		CG-Descent 5.3	1	3	2	<b>2.000e+002</b>	2.390e-007
bard	3	mBFGS	18	95	19	1.157e-001	9.765e-007
		L-CG-Descent	16	33	17	8.215e-003	3.673e-009
		L-BFGS	16	33	17	8.215e-003	3.673e-009
		CG-Descent 5.3	21	44	23	8.215e-003	1.912e-007
beale	2	mBFGS	13	76	14	4.957e-020	2.979e-010
		L-CG-Descent	15	31	16	2.727e-015	4.499e-008
		L-BFGS	15	31	16	2.727e-015	4.499e-008
		CG-Descent 5.3	18	37	19	1.497e-007	4.297e-007
biggs6	6	mBFGS	73	335	74	7.777e-013	4.920e-007
		L-CG-Descent	27	57	31	<b>5.656e-003</b>	2.514e-008
		L-BFGS	27	57	31	<b>5.656e-003</b>	2.514e-008
		CG-Descent 5.3	85	177	93	<b>5.656e-003</b>	9.195e-007
box3	3	mBFGS	21	77	22	1.692e-016	4.450e-008
		L-CG-Descent	11	24	13	3.819e-013	7.584e-007
		L-BFGS	11	24	13	3.819e-013	7.584e-007
		CG-Descent 5.3	13	27	14	1.707e-010	6.003e-007
brkmcc	2	mBFGS	4	34	5	1.690e-001	8.034e-008
		L-CG-Descent	5	11	6	1.690e-001	6.220e-008
		L-BFGS	5	11	6	1.690e-001	6.220e-008
		CG-Descent 5.3	4	9	5	1.690e-001	5.272e-008
brownbs	2	mBFGS	632	13543	633	1.952e-018	5.369e-007
		L-CG-Descent	13	26	15	0.000e+000	0.000e+000
		L-BFGS	13	26	15	0.000e+000	0.000e+000
		CG-Descent 5.3	16	40	33	1.972e-031	8.882e-010
brownden	4	mBFGS	21	312	22	8.582e+004	3.092e-010
		L-CG-Descent	16	31	19	8.582e+004	1.282e-007
		L-BFGS	16	31	19	8.582e+004	1.282e-007
		CG-Descent 5.3	38	74	48	8.582e+004	9.083e-007
chnrosnb	50	mBFGS	160	2185	161	1.263e-015	3.525e-007
		L-CG-Descent	287	564	299	6.818e-014	5.414e-007
		L-BFGS	216	427	233	1.582e-013	5.565e-007
		CG-Descent 5.3	287	564	299	6.818e-014	5.414e-007
cliff	2	mBFGS	15	75	16	1.998e-001	7.602e-008
		L-CG-Descent	18	70	54	1.998e-001	2.316e-009
		L-BFGS	18	70	54	1.998e-001	2.316e-009
		CG-Descent 5.3	19	40	21	1.998e-001	6.352e-008

cube	2	mBFGS	21	134	22	4.231e-020	6.845e-010
		L-CG-Descent	32	77	47	1.269e-017	1.225e-009
		L-BFGS	32	77	47	1.269e-017	1.225e-009
		CG-Descent 5.3	33	80	49	6.059e-015	4.697e-008
deconvu	61	mBFGS	67	855	68	1.567e-009	9.999e-007
		L-CG-Descent	475	951	476	1.189e-008	9.187e-007
		L-BFGS	208	417	209	2.171e-010	8.924e-007
		CG-Descent 5.3	475	951	476	1.184e-008	9.078e-007
denschna	2	mBFGS	7	35	8	1.468e-014	3.198e-007
		L-CG-Descent	9	19	10	3.167e-016	3.527e-008
		L-BFGS	9	19	10	3.167e-016	3.527e-008
		CG-Descent 5.3	9	19	10	7.355e-016	4.825e-008
denschnb	2	mBFGS	7	44	8	6.048e-014	4.252e-007
		L-CG-Descent	7	15	8	3.641e-017	1.034e-008
		L-BFGS	7	15	8	3.641e-017	1.034e-008
		CG-Descent 5.3	8	17	8	4.702e-014	4.131e-007
denschnb	2	mBFGS	8	55	9	1.119e-021	1.731e-010
		L-CG-Descent	12	26	14	3.253e-019	3.276e-009
		L-BFGS	12	26	14	3.253e-019	3.276e-009
		CG-Descent 5.3	12	27	15	<b>1.834e-001</b>	4.143e-007
denschnb	3	mBFGS	38	308	39	2.461e-009	3.146e-007
		L-CG-Descent	47	98	51	4.331e-010	8.483e-007
		L-BFGS	47	98	51	4.331e-010	8.483e-007
		CG-Descent 5.3	45	97	54	8.800e-009	6.115e-007
denschnf	2	mBFGS	10	75	11	4.325e-018	3.027e-008
		L-CG-Descent	8	17	9	2.126e-015	6.455e-007
		L-BFGS	8	17	9	2.126e-015	6.455e-007
		CG-Descent 5.3	11	24	13	1.104e-017	6.614e-008
djtl	2	mBFGS	79	1524	80	-8.952e+003	2.265e-002
		L-CG-Descent	82	917	880	-8.952e+003	8.865e-009
		L-BFGS	82	917	880	-8.952e+003	8.865e-009
		CG-Descent 5.3	93	770	714	-8.952e+003	3.521e-007
engval2	3	mBFGS	28	188	29	1.999e-018	9.405e-008
		L-CG-Descent	26	61	37	1.034e-016	8.236e-007
		L-BFGS	26	61	37	1.034e-016	8.236e-007
		CG-Descent 5.3	76	161	88	3.185e-014	5.682e-007
expfit	2	mBFGS	12	103	13	2.405e-001	2.916e-009
		L-CG-Descent	13	29	16	2.405e-001	4.208e-007
		L-BFGS	13	29	16	2.405e-001	4.208e-007
		CG-Descent 5.3	15	34	20	2.405e-001	1.758e-007
growthls	3	mBFGS	1	3	2	<b>3.542e+003</b>	0.000e-999
		L-CG-Descent	143	425	299	1.004e+000	3.317e-007
		L-BFGS	143	425	399	1.004e+000	3.317e-007
		CG-Descent 5.3	441	997	596	1.004e+000	1.835e-007
hairy	2	mBFGS	19	160	20	2.000e+001	6.143e-008
		L-CG-Descent	36	99	65	2.000e+001	7.961e-011
		L-BFGS	36	99	65	2.000e+001	7.961e-011
		CG-Descent 5.3	14	35	24	2.000e+001	1.044e-007
hatfldd	3	mBFGS	24	119	25	6.615e-008	1.107e-007
		L-CG-Descent	20	43	24	2.547e-007	1.936e-007
		L-BFGS	20	43	24	2.547e-007	1.936e-007
		CG-Descent 5.3	40	98	61	6.617e-008	1.934e-007
hatfldd	3	mBFGS	30	136	31	4.434e-007	6.576e-007

		L-CG-Descent	30	72	45	2.000e+001	5.012e-007
		L-BFGS	30	72	45	2.000e+001	5.012e-007
		CG-Descent 5.3	53	120	72	2.000e+001	5.012e-007
heart6ls	6	mBFGS	2266	8700	2267	2.865e-023	6.934e-009
		L-CG-Descent	684	1576	941	2.646e-010	5.562e-007
		L-BFGS	684	1576	941	2.646e-010	5.562e-007
		CG-Descent 5.3	2570	5841	3484	1.305e-010	2.421e-007
helix	3	mBFGS	22	154	23	5.489e-017	1.349e-007
		L-CG-Descent	23	49	27	1.604e-015	3.135e-007
		L-BFGS	23	49	27	1.604e-015	3.135e-007
		CG-Descent 5.3	44	90	46	2.427e-013	6.444e-007
himmelbb	2	mBFGS	1	23	2	9.665e-014	8.167e-008
		L-CG-Descent	10	28	21	9.294e-013	2.375e-007
		L-BFGS	10	28	21	9.294e-013	2.375e-007
		CG-Descent 5.3	11	23	12	1.584e-013	1.084e-008
himmelbg	2	mBFGS	7	45	8	1.070e-013	9.071e-007
		L-CG-Descent	8	20	13	9.294e-013	2.375e-007
		L-BFGS	8	20	13	9.294e-013	2.375e-007
		CG-Descent 5.3	10	24	15	1.584e-013	1.084e-008
himmelbh	2	mBFGS	7	45	8	-1.000e+000	5.026e-007
		L-CG-Descent	7	16	9	-1.000e+000	2.892e-011
		L-BFGS	7	16	9	-1.000e+000	2.892e-011
		CG-Descent 5.3	7	16	9	-1.000e+000	1.381e-007
humps	2	mBFGS	104	857	105	3.280e-016	6.351e-009
		L-CG-Descent	53	165	120	3.682e-012	8.552e-007
		L-BFGS	53	165	120	3.682e-012	8.552e-007
		CG-Descent 5.3	48	140	101	3.916e-012	8.774e-007
jensmp	2	mBFGS	1	3	2	<b>2.020e+003</b>	0.000e-999
		L-CG-Descent	15	33	22	1.244e+002	5.302e-010
		L-BFGS	15	33	22	1.244e+002	5.302e-010
		CG-Descent 5.3	13	29	19	1.244e+002	4.206e-009
kowosb	4	mBFGS	28	147	29	3.075e-004	1.367e-007
		L-CG-Descent	17	39	23	3.078e-004	3.704e-007
		L-BFGS	17	39	23	3.078e-004	3.704e-007
		CG-Descent 5.3	66	139	76	3.078e-004	8.818e-007
loghairy	2	mBFGS	74	882	75	1.823e-001	5.904e-007
		L-CG-Descent	27	81	58	1.823e-001	1.762e-007
		L-BFGS	27	81	58	1.823e-001	1.762e-007
		CG-Descent 5.3	46	136	97	1.823e-001	7.562e-008
mancino	100	mBFGS	37	1202	38	1.548e-020	1.414e-007
		L-CG-Descent	11	23	12	9.245e-021	7.239e-008
		L-BFGS	9	19	30	3.048e-021	1.576e-007
		CG-Descent 5.3	11	23	12	9.245e-021	7.239e-008
maratosb	2	mBFGS	3	59	4	-1.000e+000	5.142e-008
		L-CG-Descent	1145	3657	2779	-1.000e+000	3.216e-007
		L-BFGS	1145	3657	2779	-1.000e+000	3.216e-007
		CG-Descent 5.3	946	2911	2191	-1.000e+000	3.230e-009
mexhat	2	mBFGS	5	32	6	-4.010e-002	1.426e-012
		L-CG-Descent	20	56	39	-4.001e-002	4.934e-009
		L-BFGS	20	56	39	-4.001e-002	4.934e-009
		CG-Descent 5.3	27	61	36	-4.001e-002	3.014e-007
osborneb	11	mBFGS	53	377	54	4.014e-002	2.480e-007
		L-CG-Descent	62	127	65	4.014e-002	4.427e-007

		L-BFGS	62	127	65	4.014e-002	4.427e-007
		CG-Descent 5.3	214	423	219	4.014e-002	7.485e-007
palmer1c	8	mBFGS	32	211	33	9.760e-002	3.935e-007
		L-CG-Descent	11	26	26	9.761e-002	1.254e-009
		L-BFGS	11	26	26	9.761e-002	1.254e-009
		CG-Descent 5.3	126827	224532	378489	9.761e-002	9.545e-007
palmer2c	8	mBFGS	112	446	113	1.442e-002	8.296e-007
		L-CG-Descent	11	21	21	1.437e-002	1.257e-008
		L-BFGS	11	21	21	1.437e-002	1.257e-008
		CG-Descent 5.3	21362	21455	42837	1.437e-002	5.761e-007
palmer3c	8	mBFGS	47	245	48	1.954e-002	2.050e-008
		L-CG-Descent	11	20	20	1.954e-002	1.754e-010
		L-BFGS	11	20	20	1.954e-002	1.754e-010
		CG-Descent 5.3	5536	5777	11379	1.954e-002	9.753e-007
palmer4c	8	mBFGS	78	351	79	5.031e-002	2.235e-007
		L-CG-Descent	11	20	20	5.031e-002	3.928e-009
		L-BFGS	11	20	20	5.031e-002	3.928e-009
		CG-Descent 5.3	44211	49913	96429	5.031e-002	9.657e-007
palmer5c	6	mBFGS	13	157	14	2.128e+000	4.810e-009
		L-CG-Descent	6	13	7	2.128e+000	3.749e-012
		L-BFGS	6	13	7	2.128e+000	3.749e-012
		CG-Descent 5.3	6	13	7	2.128e+000	2.629e-009
palmer6c	8	mBFGS	56	243	57	1.639e-002	6.900e-007
		L-CG-Descent	11	24	24	1.639e-002	5.520e-009
		L-BFGS	11	24	24	1.639e-002	5.520e-009
		CG-Descent 5.3	14174	142228	28411	1.639e-002	7.738e-007
palmer7c	8	mBFGS	41	212	42	6.020e-001	5.201e-007
		L-CG-Descent	11	20	20	6.020e-001	7.132e-009
		L-BFGS	11	20	20	6.020e-001	7.132e-009
		CG-Descent 5.3	65294	78428	149585	6.020e-001	9.957e-007
palmer8c	8	mBFGS	48	361	49	1.598e-001	1.099e-009
		L-CG-Descent	11	18	17	1.598e-001	2.376e-009
		L-BFGS	11	18	17	1.598e-001	2.376e-009
		CG-Descent 5.3	8935	9903	19183	1.598e-001	9.394e-007
rosenbr	2	mBFGS	32	241	33	1.383e-016	4.603e-007
		L-CG-Descent	34	77	44	4.691e-018	7.167e-008
		L-BFGS	34	77	44	4.691e-018	7.167e-008
		CG-Descent 5.3	37	86	52	1.004e-014	1.894e-007
sineval	2	mBFGS	69	489	70	1.910e-019	1.168e-008
		L-CG-Descent	60	143	87	1.556e-023	1.817e-011
		L-BFGS	60	143	87	1.556e-023	1.817e-011
		CG-Descent 5.3	62	172	122	1.023e-012	5.575e-007
sisser	2	mBFGS	19	83	20	3.860e-010	4.587e-007
		L-CG-Descent	6	18	14	6.830e-012	2.220e-008
		L-BFGS	6	18	14	6.830e-012	2.220e-008
		CG-Descent 5.3	6	13	7	3.026e-014	3.663e-010
tointqor	50	mBFGS	39	615	40	1.176e+003	4.033e-007
		L-CG-Descent	29	36	53	1.175e+003	4.467e-007
		L-BFGS	28	35	51	1.175e+003	7.482e-007
		CG-Descent 5.3	29	36	53	1.175e+003	4.464e-007
vardim	200	mBFGS	22	154	23	1.237e-021	1.376e-009
		L-CG-Descent	10	21	11	4.168e-019	2.582e-007
		L-BFGS	7	31	27	5.890e-025	3.070e-010

		CG-Descent 5.3	10	21	11	4.168e-019	2.582e-007
watson	12	mBFGS	61	308	62	1.130e-008	3.081e-007
		L-CG-Descent	49	102	54	1.592e-007	8.026e-007
		L-BFGS	48	97	49	9.340e-008	1.319e-007
		CG-Descent 5.3	726	145	727	1.139e-007	8.115e-007
yfitu	2	mBFGS	73	462	74	6.670e-013	1.938e-007
		L-CG-Descent	75	177	106	8.074e-010	3.910e-007
		L-BFGS	75	177	106	8.074e-010	3.910e-007
		CG-Descent 5.3	147	327	189	2.969e-011	5.681e-007

We summarize the comparison of the test results as follows:

1. For two problems (`arglina` and `biggs6`), `mBFGS` converges to better points than `L-CG-Descent`, `L-BFGS`, and `CG-Descent 5.3`. For another 2 problems (`growthls` and `jensmp`), `L-CG-Descent`, `L-BFGS`, and `CG-Descent 5.3` converge to better points.
2. For 19 problems, `mBFGS` converges faster than `L-CG-Descent`, `L-BFGS`, and `CG-Descent 5.3`. For about 10 problems, `mBFGS` converges slower than `L-CG-Descent`, `L-BFGS`, and `CG-Descent 5.3`. For the rest problems, `mBFGS` converges either faster than some but slower than other codes or as faster as all other codes.

Based on these numerical test results, we believe that the proposed algorithm is very promising.

## 5 Conclusions

We have proposed a modified BFGS algorithm and proved that the modified BFGS algorithm is globally and superlinearly convergent. We have shown that the computational cost in each iteration is almost the same for both the BFGS and the modified BFGS. We have provided numerical test results and compared the performance of the modified BFGS to the performance of implementations of other established and state-of-the-art algorithms, such as BFGS, limited memory BFGS, descent and conjugate gradient, and limited memory descent and conjugate gradient. The results and comparison show that the modified BFGS algorithm appears very effective.

## 6 Acknowledgements

The author would like to thank Professor Jorge Nocedal who suggested conducting the test for the modified BFGS and BFGS against the CUTE test problems, and comparing the performance. The author also thanks Dr. Sven Leyffer at Argonne National Laboratory for making him aware of Professor W. Hager's website which has test results of `L-CG-Descent`, `L-BFGS`, and `CG-Descent 5.3`. The author is grateful to Professor Teresa Monterio at Universidade do Minho for providing the software to convert AMPL mod-files into nl-files, which makes the test possible.

## References

- [1] <http://www.orfe.princeton.edu/rvdb/ampl/nlmodels/index.html>.
- [2] D.P. Bertsekas (1996), *Nonlinear Programming*, Belmont, Massachusetts: Athena Scientific.
- [3] I Bongartz and A R. Conn and N. Gould and P.L. Toint (1995), CUTE: Constrained and Unconstrained Testing Environment, *ACM Transactions on Mathematical Software*, Vol. 21, pp. 123-160.
- [4] Yu-Hong Dai (2002), Convergence properties of the BFGS Algorithm, *SIAM Journal of optimization*, Vol. 13, pp. 693-701.

- [5] G. H. Golub and C. F. Van Loan (1989), *Matrix Computations*, Baltimore: The Johns Hopkins University press.
- [6] W. W. Hager and H. Zhang, <http://www.math.ufl.edu/hager/CG/results6.0.txt>.
- [7] W. W. Hager and H. Zhang, *The limited memory conjugate gradient method*, <http://www.math.ufl.edu/hager/CG/results6.0.txt>.
- [8] W. W. Hager and H. Zhang (2005), A new conjugate gradient method with guaranteed descent and an efficient line search, *SIAM J. Optimization*, Vol. 16, pp. 170-192.
- [9] Dong-Hui Li and Masao Fukushima (2001), A modified BFGS method and its global convergence in nonconvex minimization, *Journal of Computational and Applied Mathematics*, Vol. 19, pp. 15-35.
- [10] J. More and D. J. Thuente (1990), On line search algorithms with guaranteed sufficient decrease, *Technical Report MCS-P153-0590*, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL.
- [11] J. Nocedal (1980), Updating quasi-Newton matrices with limited storage, *Math. Comp.*, Vol. 35, pp. 773-782.
- [12] J. Nocedal and S.J. Wright (1993), *Numerical Optimization*, New York: Springer-Verlag.
- [13] M. J. D. Powell (1976), Some global convergence properties of a variable metric algorithm for minimization without exact line searches, In R. W. Cottle and C. E. Lemke, (Eds.) , *SIAM-AMS Proceedings Vol. IX*, Philadelphia, PA: SIAM, pp. 53-72.
- [14] Z. Wei and G. Li and L. Qi (2006), New quasi-Newton methods for unconstrained optimization problems, *Applied Mathematics and Computation*, 175, pp. 1156-1188.
- [15] P. Wolfe (1969), Convergence conditions for ascent methods, *SIAM Review*, Vol. 11, pp. 226-235.
- [16] P. Wolfe (1971), Convergence conditions for ascent methods II: Some Corrections, *SIAM Review*, Vol. 13, pp. 185-188.
- [17] Y. Yang (2012), A Globally and Quadratically Convergent Algorithm for Nonconvex Unconstrained Optimization, *arXiv:1212.5452 [math.OC]*.
- [18] G. Zoutendijk (1970), Nonlinear Programming, Computational Methods, In J. Abadie, (Eds.) , *Integer and Nonlinear Programming*, North Holland: Amsterdam, pp. 37-86.